Automated Reading of High Volume Water Meters

by

Jessica Ulyate

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering at Stellenbosch University

Supervisor: Dr. R. Wolhuter
Department of Electrical and Electronic Engineering

March 2011

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2011

Abstract

Accurate water usage information is very important for municipalities in order to provide accurate billing information for high volume water users. Meter reading are currently obtained by sending a person out to every meter to obtain a manual reading. This is very costly with regards to time and money, and it is also very error prone.

In order to improve on this system, an image based telemetry system was developed that can be retrofitted on currently installed bulk water meters. Images of the meter dials are captured and transmitted to a central server where they are further processed and enhanced. Character recognition is performed on the enhanced images in order to extract meter readings.

Through tests it was found that characters can be recognised to 100% accuracy for cases which the character recognition software has been trained, and 70% accuracy for cases which is was not trained. Thus, an overall recognition accuracy of 85% was achieved. These results can be improved upon in future work by statistically analysing results and utilizing the inherent heuristic information from the meter dials.

Overall the feasibility of the approach was demonstrated and a way forward was indicated.

Samevatting

Dit is belangrik vir munisipaliteite om akkurate water verbruikingssyfers te hê sodat hulle akkurate rekeninge aan hoë volume water gebruikers kan stuur. Tans besoek 'n persoon fisies elke meter om meterlesings te verkry. Dit is egter baie oneffektief ten opsigte van tyd en geld. Die metode is ook baie geneig tot foute.

Ten einde te verbeter op hierdie stelsel was 'n beeld gebaseerde telemetrie stelsel ontwerp wat geïnstalleer word op huidig geïnstalleerde hoë volume water meters. Beelde van die meters word na 'n sentrale bediener gestuur waar dit verwerk word en die beeld kwaliteit verbeter word. Karakter herkenning sagteware word gebruik om die meter lesings te verkry vanuit die verbeterde beelde.

Deur middel van toetse is gevind dat karakters herken kan word tot op 100% graad van akkuraatheid in gevalle waar die karakter herkenning sagteware opgelei is, en 70% akkuraatheid vir gevalle waarvoor dit nie opgelei was nie. Dus was 'n algehele herkennings akkuraatheid van 85% behaal. Hierdie resultate kan verbeter word in die toekoms deur die resultate statisties te analiseer en die inherente heuristieke inligting van die meter syfers te benutting.

Ten slotte, in die tesis was die haalbaarheid van die benadering gedemonstreer en 'n weg vorentoe vir toekomstige werk aangedui.

Acknowledgements

I would like to express my sincere gratitude towards the following people:

- My promoter, Dr. R. Wolhuter for his guidance and insight as well as the many hours spent reviewing my work.
- Prof J. du Preez for his insights and invaluable suggestions.
- My friends and family for all their encouragement throughout this project.

Contents

List of Figures

List of Tables

List of Appreviations

CMOS	Complementary Metal Oxide Semiconductor
FTP	File Transfer Protocol
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
LED	Light Emitting Diode
OCR	Optical Character Recognition
OpenCV	Open Computer Vision
UART	Universal Asynchronous Receiver/Transmitter

Chapter 1

Introduction

1.1 Motivation

The project was prompted by a request from Cape Town Municipality for a more efficient way obtain bulk water meter readings. Accurate meter readings are required by the Municipality in order to accurately bill water usage, especially where large quantities of water are involved. The current system involves sending a person out into the field to every meter to obtain a manual reading. This method is very costly with regards to time and money and, according to information received via the user organisation, also error prone . It was requested that a device be built capable of obtaining meter readings automatically, that can be conveniently retrofitted on currently installed analogue water meters. Digital flow meters are available, but they are not practical to install as water flow has to be interrupted during installation, stricter requirements are imposed on the surrounding pipeline geometry and the costs involved are high. Digital meters also require electrical infrastructure and a means to capture, store and transmit flow data. This places restrictions on where they can be installed and renders them less versatile than analogue mechanical water meters.

It was, therefore, decided to follow a process where existing infrastructure is retained

unaltered, but retrofitted with an electronic capture device.

1.2 Objectives

The objectives of the work documented in this thesis, are as follows:

- Develop a standardised, cost effective image capturing device that can be retrofitted on currently installed bulk water meters, capable of transmitting images to a central server.
- Automatically process images and perform character recognition on them in order to extract the meter readings.
- Determine the level of extraction accuracy that could be achieved with relative ease, in a reasonably short time without resorting to computationally very complex and time consuming methods.

1.3 Operational Contributions

The following enhancements to the existing operational infrastructure were realised:

- An image based telemetry system that utilizes the current GSM infrastructure and which can be retrofitted on currently installed bulk water meters, was developed.
- Dedicated camera driver software was developed for a Wavecom chip as well as a computer.
- Open-source character recognition programs were evaluated to find a program possibly suitable for the application.
- Image enhancement algorithms were developed in order to improve the accuracy of the character recognition software.

- A custom region growing algorithm was developed that is specifically suited to this application.
- The character recognition software was trained specifically for bulk water meter dials.
- Image enhancement techniques were refined until an acceptable level of character recognition accuracy was achieved.

1.4 Overview

In chapter 2 a brief outline is presented of the proposed steps for implementing the project. The project is divided into 3 main parts, and each part is discussed separately.

A literature study is performed in chapter 3. Design considerations with regards to bulk water meters are discussed. The main focus of this chapter, however, is the evaluation of character recognition software. The different open-source character recognition programs are evaluated in order to find the one most suited to the application.

In chapter 4 the implementation outline is presented. The details of how each part of the program was implemented, is explained and examples given where possible.

An in depth discussion of key program components is performed in chapter 5 in order to facilitate a better understanding of the how the components work.

The steps that were involved in reaching the final implementation of the program components are discussed in chapter 6. Each step implementation is explained together with the reason why it did not produce results suitable for the project. The aim of this chapter is to show the motivation behind the selection of each program component.

An experimental investigation is performed in chapter 7 in order to determine the accuracy of the character recognition software with the enhanced images. It was shown that a recognition accuracy of 100% can be achieved for cases that the character recognition

software has been trained for, and 70% recognition accuracy for cases that is has not been trained for. These results are then discussed and summarised.

The final chapter summarises the document and provides an overview of the project and operational contributions. The chapter concludes by recommending future work to be done on the project.

Chapter 2

Proposed Methodology

2.1 Introduction

This chapter covers the proposed project implementation. The objective is to briefly describe the project components and what steps are planned to realise these components. This will be done by dividing the project into three main parts and discussing each part separately.

2.2 Hardware

A device that can capture images from water meters before transmitting them to a server will be developed. The GSM network will be used for communications utilizing the current cellular infrastructure. A digital camera module will be used for the image capturing and the device will be mounted in a standardized fixture with adequate light sources.

2.3 Software

It is proposed that software be developed, or adapted from an existing, suitable, open source application, if available, and executed in a Linux environment. The image taken by the camera contains the meter dials as well as other surrounding parts of the meter. These other parts will hinder character recognition, therefore, only the numbers on the meter dials need to be extracted from the image. The first step would be to enhance the image to improve character recognition results. After the image has been enhanced, the numbers have to be extracted to produce a clear image for the character recognition software. When a clear black and white image containing the numbers has been produced, the image is then ready to be passed to a character recognition program.

2.4 Character Recognition

Recognising characters from an analogue water meter is unlike normal text, as numbers can be visible in various stages as the dials roll between numbers. Character recognition software has to be found that can easily be trained for a custom character set and thus be trained to recognise partially visible numbers. The character recognition software will then be trained with sufficient samples of numbers in various stages and the recognition results evaluated in terms of accuracy.

2.5 Conclusion

This chapter briefly described what steps are proposed in order to implement the project. The next chapter will cover the necessary background and literature studies performed in order to determine the feasibility of this project.

Chapter 3

Literature Study

3.1 Introduction

This literature study has been undertaken to investigate which considerations should go into important design decisions in the project. It was important to decide how bulk water meter readings would be retrieved and once that was decided, how to carry out character recognition. The first objective of this chapter is to determine the options available for automatically retrieving readings from bulk water meters. The second objective is to determine which character recognition options are available and to find a suitable option for this application. This will be achieved by determining which character recognition software characteristics are important for this project and evaluating the available software accordingly.

3.2 Bulk Water Meters

As stated in the project motivation, it is very important to have accurate readings of the bulk water meters for billing purposes. Analogue mechanical meters and digital meters are available, as well as digital add-on devices for mechanical meters.

The Proline Promag 10H is one of the basic digital flow meters available from Endress & Hauser. The flow meter has to be installed in a section of pipeline[1], as can be seen in figure 3.1, thus, a section of pipeline has to be modified. It is also required to shut off the water during installation, which is not acceptable to all bulk water users. As with other digital meters, the Proline Promag 10H requires a power source in order to perform its main function of recording water usage. It is not always possible to supply digital meters with a constant source of electricity due to installations in remote locations. Even if electricity is available, unforeseen power outages are a reality in South Africa which can affect the accuracy of readings. Mechanical meters are not affected by power outages.

Figure 3.1: The Proline Promag 10H digital flow meter.

Digital devices that can be mounted on mechanical meters that count optical pulses or observe rotating arms are available. They do not provide the actual meter reading, but rather how much water has been used since they have been installed. The OPTO Pulser from Meinecke Meters is such as device, and it is compatible with the bulk water meters used in this project. Figure 3.2 shows one installed on a bulk water meter. The Pulser requires a power source[2] as well as an additional hardware interface to monitor the pulse outputs. From information received via the user organisation, there are inevitably problems with missed pulses or with communication with the devices. Thus, their accuracy

in extracting accurate meter readings is brought into question. Thus, mechanical meters are still the most reliable way to keep track of water usage.

Figure 3.2: The OPTO Pulser installed on a bulk water meter.

Thus, it is more feasible to develop a more reliable add-on device for mechanical meters than installing digital meters. The proposed device should be a convenient addition to mechanical meters without disrupting their normal operation. It should capture images of the meter dials and save the images for later reference. The device should be mounted so that the dials on the meter could still be read manually, if the device is unavailable. The issues with power sources with the digital water meters are not relevant to the proposed device as it is not the primary usage monitoring tool, if it does not function metering is not affected.

3.3 Character Recognition Software

To determine the values on the bulk water meters, a suitable character recognition program is required. The investigation of possible suitable character recognition software is

the most important part of the Literature Study. Typical required characteristics are as follows:

- The user must be able to train a custom character set for the recognition software.
- The software has to be open source.
- The software must be compatible with Linux.
- The software must be well maintained.
- The software must support command line execution.

3.3.1 Requirements

3.3.1.1 Trainable

The meter dials advance by rotation, thus at any point a number can be at any stage of partial visibility. Character recognition programs are designed to recognize scanned text, thus they are only trained to recognize whole characters. In this application it is important to recognize partial numbers as well, thus a suitable program must have the option to train a custom character set for the recognition software.

3.3.1.2 Open Source

This project is undertaken by a student as a Masters topic, thus a limited budget is a reality. Part of the study's objective is to find the cheapest possible solution to the problem. Open source software is free to use and as the source code is available for inspection, it is possible to gain a better understanding of the program's functionality.

3.3.1.3 Linux Compatible

Linux is an open source, licence free operating system. This characteristic ties in with the monetary concerns addressed in the previous section.

3.3.1.4 Well Maintained

The software must be maintained on a regular basis. This makes it less likely that the program contains errors or bugs. It also makes it more likely that there is a community of people actively working with the software that can provide support via forums and message boards during development.

3.3.1.5 Command line Execution

It is important that the software supports command line execution in order to facilitate end product automation. Ideally the entire process of retrieving meter readings should be automated: from taking the photos, to processing the images, to displaying the character recognition results. A program with only a graphical user interface would provide a substantial problem in this regard.

3.3.2 OCR Methods

There are two main methods of implementing character recognition in software; Matrix Matching and Feature Extraction.

- In Matrix Matching, the unidentified character is compared to a library of character matrixes or templates. The unidentified character is classified as the character whose template it matches the closest[3].
- In Feature Extraction the software looks at general features of characters, like lines, open spaces and line intersections to identify unknown characters[3].

3.3.3 Evaluation

The following character recognition programs were investigated:

- GOCR

- Ocrad
- Tesseract
- ABBYY FineReader
- CuneiForm

3.3.3.1 GOCR

GOCR was first released in December 2000 with the latest release in September 2010[4]. GOCR is a command line OCR program and is also known as JOCR (Jorg's Optical Character Recognition) as GOCR was already registered on the online code repository SourceForge. GOCR employs the feature extraction method for recognition. According to the README file[4], the program is still in its early stages. For good results, the image scans need to be clear with well separated characters. Noisy images, slanted fonts and rotated images do not work well with the program. There is no mention of being able to train the program for a different character set. Overall the documentation is limited and the software presents a number of obstacles which makes implementation difficult.

3.3.3.2 Ocrad

Ocrad was first released in 2003 with the latest release in July 2010[5]. Ocrad can be used as a backend for other programs, or as a stand alone OCR program. Ocrad employs the feature extraction method for recognition[5]. According to the Ocrad online manual[5], it is mainly a research project that is subject to change as the authors understanding of OCR progresses. It also states that it is very sensitive to character defects and it is difficult to modify the program to recognize new characters.

3.3.3.3 Tesseract

Tesseract was first developed as proprietary software for Hewlett-Packard between 1985 and 1995. Ten years passed without any development taking place and in 2005 it was released as open source software that is currently being developed by Google[6] [7]. Tesseract is a command line OCR program. Tesseract employs feature extraction to identify characters. In 1995 Tesseract was one of the top three character recognition engines in the UNLV (University of Nevada Las Vegas) Fourth Annual Test of OCR Accuracy[8]. Tesseract does not support page layout analysis, thus it cannot interpret multi column text and it only supports uncompressed tiff input files. It is possible though, to train a new character set for the program[9] and the documentation provided is clear and extensive.

3.3.3.4 ABBYY FineReader

ABBYY was founded in 1989, specialising in text recognition technologies and applied linguistics. FineReader is proprietary software aimed at the commercial market, and was designed to convert scanned images into editable text[10]. As the software is proprietary, it is unclear which OCR method it employs for recognition. It is reviewed regularly throughout the world, and consistently receives good recommendations. Overall, the software seems more suited to processing documents. As the software is proprietary, it is not suitable for this application.

3.3.3.5 CuneiForm

CuneiForm was initially developed in 1996 as proprietary software for Cognitive Technologies. After no new developments took place for a few years, it was released as freeware in 2007, and the kernel released as open-source in 2008[11]. The latest version was released in 2009. It is unclear which OCR method CuneiForm uses. CuneiForm can recognize any printed font, but is not suitable for recognizing handwriting. Cognitive Technologies is

still involved with the software, and since the company is Russian, most of the information available on CuneiForm is in Russian which limits the availability of documentation.

3.3.4 Conclusion

Tesseract is the only software that can be trained for a new character set, which is vital for this project. Its inability to process multi column text is not a problem in this application, as only a single line of numbers needs to be recognized. The input file format limitation is not restrictive as converting an image file format is a trivial operation when using the Linux command line.

3.4 Tesseract In Depth

In this section, the working of Tesseract is discussed in greater details.

3.4.1 Recognition

When Tesseract is trained for a new character set, polygonal approximation is used to save the features of the new character. During the character recognition process, small fixed length features are extracted from the outline of the unidentified character. These features are then matched many-to-one against the features of the training data.

The small feature matching has the advantage that broken characters can easily be handled. It does however mean that the computational cost of the algorithm is higher due to the computing of the distance between the unidentified character and the known features. This does not pose a problem as the software is executed on a dedicated central server which can provide ample resources.

3.4.2 Recognition Process

In Tesseract, the character recognition process works as follows:

- Firstly, text blobs are found and the outlines stored.
- The blobs are then organized into text lines.
- The text baselines are found. These lines are used to correct skew text which can occur in scanned images or warped text which occurs where pages are attached to a book spine.
- The text blobs are then analysed to determine whether the letters are fixed pitch, in other words spaced evenly, or proportional pitch.

  - When the text is fixed pitched the pitch can be determined easily and the letters extracted.

  - With proportional pitch, letter extraction becomes a non-trivial process involving chopping of joined characters and joining of broken characters.
- Word recognition is performed in two passes. An adaptive classifier is used in the process which 'learns' as the recognition progresses. Since the classifier knows more at the end of the text, the text is processed a second time in order to benefit from the new classifier knowledge.

3.4.3 Components

3.4.3.1 Character Classifier

The character classifier uses small feature matching to large prototypes in order to find unidentified characters. During classification, a shortlist of possible character classes is assembled. The possible characters are filtered and the similarities between them and

the unknown character are computed with a sum of product expression. This is then evaluated to find the best match.

3.4.3.2 Linguistic Analysis

Tesseract employs very little linguistic analysis in the recognition process. The following six categories are maintained and consulted when word segmentation is performed:

- Top Frequent Word
- Top Dictionary Word
- Top Numeric Word
- Top UPPER case Word
- Top lower case Word
- Top Classifier Choice

Each of the above categories is multiplied with a different constant and the word with the lowest total distance rating is selected.

3.4.4 Conclusion

Since baseline fitting is performed on the text passed to Tesseract, it will not be possible to pass individual half-numbers to Tesseract for recognition as the two halves will be detected as two separate text blobs. Thus all the dial numbers have to be passed to Tesseract in one image.

Also, when passing numbers to Tesseract, ensure that they are fixed-pitch as this reduces computational complexity by eliminating many of the complex steps associated with proportional pitch.

3.5 Conclusion

In this chapter it was shown that it makes more sense to develop an add-on device for analogue bulk water meters than replacing them with digital meters because of monitory factors and reliability considerations.

In was also concluded that Tesseract would be the best software to use for character recognition as it is the most versatile and well maintained software available.

Upon studying Tesseract closer, the best way to pass characters to Tesseract was determined.

The next chapter will expand on the methods proposed in this chapter and explain the details of how they were implemented.

Chapter 4

Implementation Outline

4.1 Introduction

This chapter will expand on the Proposed Methodology chapter and detail how the proposed methods were implemented. The objective is to explain in detail the steps involved in obtaining a meter reading from the photos of bulk water meter dials. This will be done by dividing the process of obtaining meter readings into four main sections. A detailed description of steps involved in each section will be discussed. Where relevant, the reader will be referred to other chapters in this document for a more in depth discussion of some concepts. All methods discussed here were implemented with Python and executed in Linux.

4.2 Image Procurement

Image procurement encompasses the process of gathering images used for the image enhancement and character recognition processes. A general overview of the process is illustrated in fig 4.1. Image procurement can be split into two parts; image capturing and image transmission.

- Image capturing is the process of taking a photo of the bulk water meter dials. It is done with the C328 COMedia UART camera which is attached to the Wavecom Q2686 processor.

- Image transmission is the process of moving the captured image from the field device to a central server. The Wavecom Q2686 processor has a built in GSM modem which is used for this function.

Both of these functions are performed on one device which consists of the Wavecom Q2686 development board with the camera attached to it.

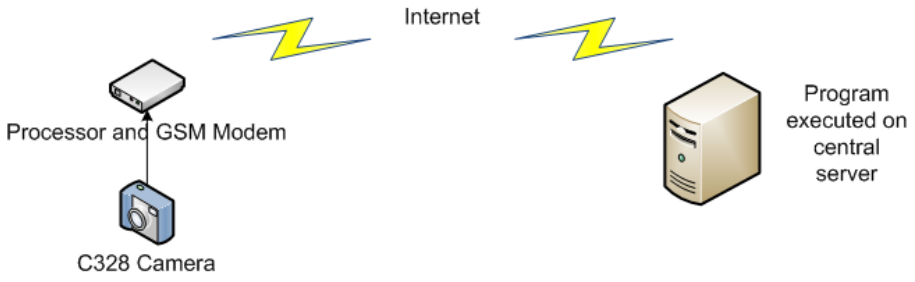


Figure 4.1: The process of transmitting the image from the bulk water meter to the central server.

The process is illustrated in fig 4.2 and happens as follows:

- When the device is switched on, the Wavecom GSM services are first initialised. This includes initialising the IP stack which is used for FTP operations.

- Next, communication with the camera is established and the parameters for image capturing are initialised.

- When the camera initialisation is finished, a command is issued to the camera to capture an image. The camera module has to acknowledge the command for the process to continue.

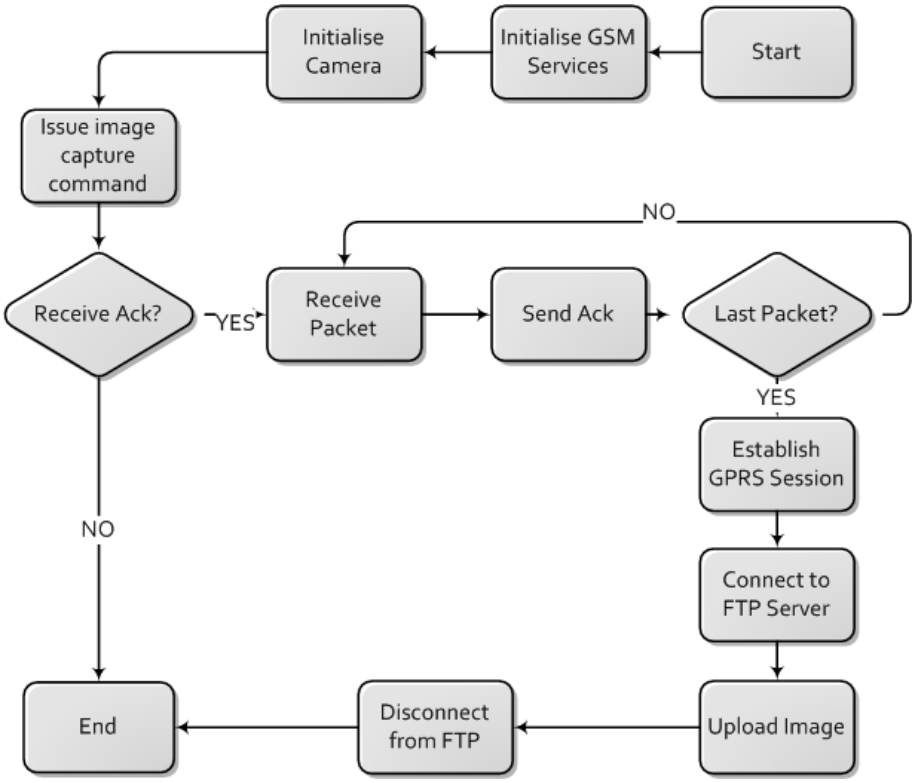


Figure 4.2: The process of capturing an image and uploading it from the device to an FTP server.

- Once the acknowledge has been received, the processor start a cycle of waiting for image packets from the camera module and acknowledging the receipt of each packet until the last image packet has been received.
- When the image is complete, the processor establishes a GPRS connection and logs on to a specified FTP server to where it uploads the image.
- Upon completion of the image upload the FTP and GPRS sessions are terminated.

4.3 Rectangle Detection

Rectangle detection is performed to find the coordinates of the window containing the dials. To aid in the correct identification of the window, a high contrast border is placed around it. To find rectangles in the image, OpenCV example code is used which returns a list of the coordinates of all rectangles found in the image. The rectangle with the smallest area is then selected for the dials window. The window is cropped and then saved as a new image.

The OpenCV rectangle detection code finds rectangles by running an edge detection algorithm on the image while performing a threshold with various values. Once this is done, contours are approximated with polygons from the edges found. If an approximation has four corners and the corner angles are close to 90 degrees, the approximation is classified as a rectangle.

During development the rectangle detection code performed so well, it was never deemed necessary to test it further as the success rate was almost 100%. In the rare cases which it did return the wrong rectangle, it was because of lighting errors such as shadows occurring on the meter face.

The meter with a high contrast border is shown in figure 4.3 and the cropped window in figure 4.4.

Figure 4.3: High contrast border around meter dials.

Figure 4.4: Cropped image after rectangle detection.

4.4 Image Enhancement

In this step, the image is converted from greyscale to black and white. It is important to do the conversion with as little as possible loss of quality to the structure of the numbers.

4.4.1 Smoothing

The image is smoothed to preserve number quality in later steps. Images of the dials before and after smoothing are shown in figure 4.6 and figure 4.7. A median filter is used for smoothing, as it reduces noise and preserves edges in the image. A median filter works by replacing a pixel value with the median values of all the neighbouring pixels within a specified window[12]. A median filter with a 3x3 window is applied to the image with the help of the OpenCV library. An example of a median filter is shown in figure 4.5. The median filter smoothed out some irregularities around the digits in the image. It also evened out the background color, thus increasing the number contrast slightly. Artefacts in the details are also smoothed out, such as the lines between the numbers, which appear less pronounced in the second image. The advantages of using a median filter are much more apparent after the image has been thresholded. In section 7.4, figure 7.2 shows a thresholded image where a median filter was applied in the previous step, figure 7.3 shows a thresholded image where the median filter was not applied in the previous step.

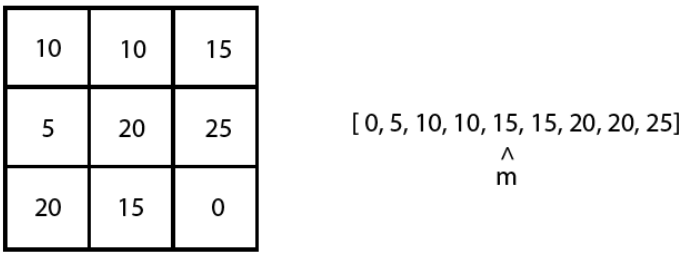


Figure 4.5: Example of a 3x3 median filter. The m indicates the median value.

Figure 4.6: Image before applying smoothing.

Figure 4.7: Image after applying smoothing.

4.4.2 Histogram Modification

To increase the brightness and the contrast of the image, histogram modification is performed. Due to lighting, some white areas on the dials may appear off-white and some black areas may appear dark grey. The goal of this process is to correct these colour flaws in the image by manipulating the histogram of the image. Histogram modification is discussed further in section 5.3.

4.4.3 Threshold

Thresholding is a method for producing black and white images. Image pixel values range between 0 and 255, where a value of 0 represents black and a value of 255 represents white. In thresholding, a threshold value is chosen, all pixels with a value less than the threshold are set to black and all pixels with a value greater than the threshold are set to white[13]. Through visual inspection and some empirical experimentation a suitable threshold value was found for the meter images. Images of dials before and after thresholding are shown in figure 4.8 and figure 4.9.

Figure 4.8: Image before applying thresholding.

Figure 4.9: Image after applying thresholding.

4.5 Character Extraction

In order to produce a clean, noise free line of numbers for the character recognition software, the individual numbers from the dials need to be extracted from the image.

4.5.1 Region Growing

A region growing algorithm is used to find the coordinates of the numbers. The algorithm works by stepping through a line of horizontal pixels. When a dark pixel is encountered, the algorithm finds all other dark pixels that are directly connected to it. The coordinates of the rectangle containing all the dark pixels, is recorded as a region. When the algorithm is finished it returns a set of regions. To make sure that regions aren't missed, the algorithm is run on two horizontal pixel lines at different vertical positions in the image. Region growing lines, as well as regions found are illustrated in figure 4.10. The details of the region growing algorithm is discussed in section 5.4.

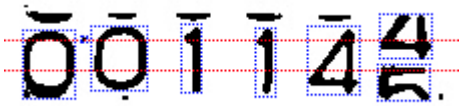


Figure 4.10: Meter dials with two region growing lines illustrated as well as regions found.

4.5.2 Filtering

The region growing step produces two sets of regions that need to be compared in order to find corresponding regions. The sets also need to be filtered to remove regions that do not contain numbers. The following steps are executed:

- Remove any regions that have an area that is smaller than a specified value. This eliminates small regions that can be considered as noise.
- Find identical regions in the two sets and add them to a new, final set.
- Find regions that are different, but lie on the same vertical line. Create a new region that contains both of the previous regions and add it to the final set. This

ensures that when the halves of two numbers are visible on the meter dial, they are considered as one unit.

- Compare the regions in the final set with a set of coordinates representing the approximate centrelines of the meter dials. Remove regions that do not correspond with any centrelines.

- Crop regions to the approximate dial width. This removes any dark regions that could originate from shadows cast on the meter face.

The methods used for determining the centrelines and dial widths, as well as how the filtering process in implemented is discussed further in section 5.5 and section 5.6.

4.5.3 Extraction and Construction

The coordinates from the final set of regions are used to extract the areas in the image containing numbers and then save them in separate images. The separate images are then combined to form a new image that contains a clear straight line of black and white numbers. An image of the dials before extraction is shown in figure 4.11, and a reconstructed image in figure 4.12.

Figure 4.11: Image before extracting numbers.

Figure 4.12: Reconstructed image.

4.6 Character Recognition

Tesseract is the program used for character recognition. To recognize meter readings accurately, Tesseract was trained with images of whole numbers and images of the halves of two different numbers, hereafter referred to as half-numbers.

4.6.1 Character Definitions

In order to differentiate between whole numbers and half-numbers, Tesseract is trained to output different values for each case. Tesseract is trained to output a letter when a half-number is found. Table 4.1 lists the half-numbers and their corresponding letters.

4.6.2 Training

4.6.2.1 Gather Samples

Photos were taken of the meter dials in various positions. The images were subjected to the image enhancement techniques discussed earlier in this chapter. Five different images of each number and half-number were then collected. The same meter was used in all the photos, as the font remains the same between two meters of the same model.

4.6.2.2 Construct Image

The samples were combined into one image file used for the training. The samples were arranged in multiple lines in a random order in the image file. Part of the training file is

Table 4.1: Half-numbers and the corresponding letters that Tesseract outputs when the number is found.

Half-number	Corresponding Letter
01	a
12	b
23	c
34	d
45	e
56	f
67	g
78	h
89	i
90	j

shown in figure 4.13.

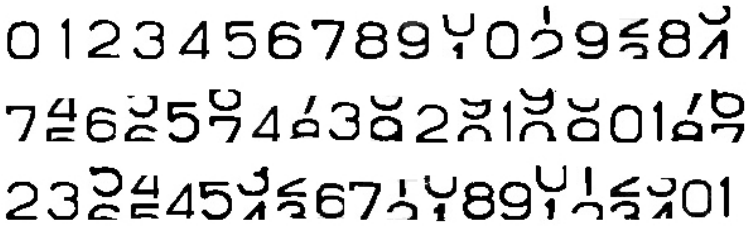


Figure 4.13: A part of the training image used for Tesseract.

4.6.2.3 Create Character Set Files

The training image is first run through a program that finds the bounding boxes for each character. The user must then manually correct the boxes and identify the character in

the box. Next, a series of commands are executed to create all the character set files. The files must then be copied to the Tesseract folder containing character set data. The process of creating the character set files are described in detail on the Tesseract training webpage [9].

4.6.3 Using Tesseract

Tesseract is executed via a single command from the command line. In the command, the input image, the output file and the character set to be used for recognition are specified. The success of Tesseract is discussed later in the Experimental Investigation chapter.

4.7 Conclusion

In this chapter the steps involved in obtaining a reading from images of bulk water meters were grouped into four main sections. The methods employed in each section were elaborated on and other sections in this document were referenced for more advanced concepts.

In the next chapter the issues encountered while implementing these methods will be discussed.

Chapter 5

Detailed Work

5.1 Introduction

The objective of this chapter is to explain the functioning of important program components in detail. Each component will be discussed separately and explained with the help of images and flow diagrams where possible.

5.2 Image Capturing

5.2.1 Physical Setup

As mentioned earlier in the Implementation Outline chapter, images are captured with the C328 COMedia UART camera which is attached to the Wavecom Q2686 processor. The block diagram of the connections are shown in figure 5.1. A TTL level converter was necessary for communication between the camera and the processor as the logic voltage levels differed between the two devices. The camera input voltage is also different from the processor input voltage. In order to simplify the setup, a voltage converter was used with the processor power source in order to provide a different voltage for the camera.

Everything was build on the Wavecome Q2686 Development Board. Figure 5.2 shows the physical setup of the block diagram with a close-up of the development board in figure 5.3.

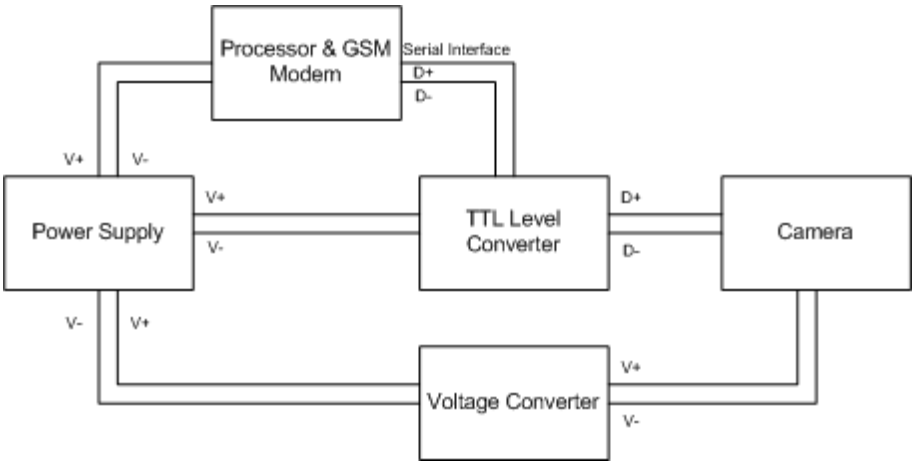


Figure 5.1: A block diagram showing how the COMedia UART camera is connected to the Wavecom processor.

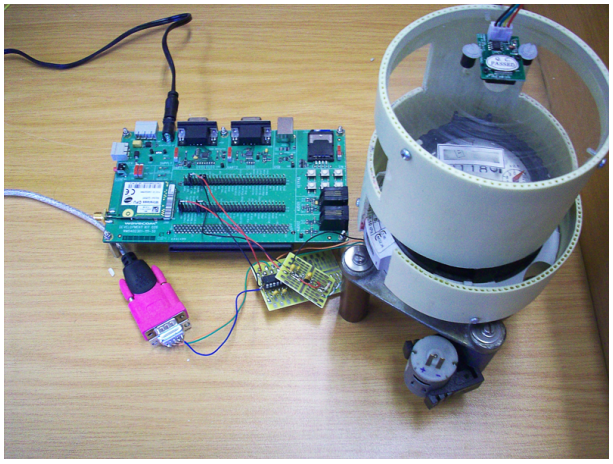


Figure 5.2: The physical setup of the block diagram in figure 5.1.

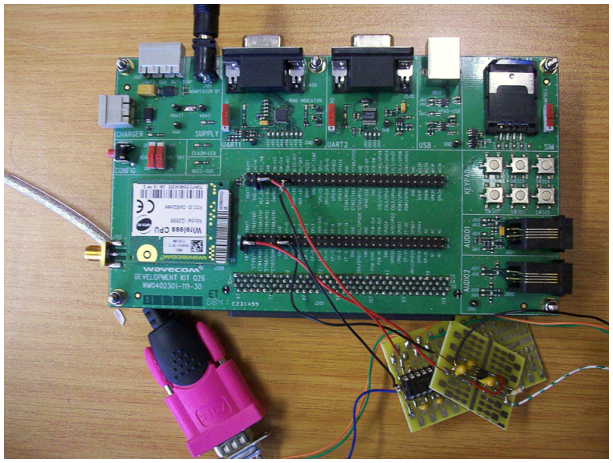


Figure 5.3: A close-up image of the development board.

5.2.2 Hardware Selection

The Wavecom development board with processor was sponsored, thus the project evolved with that specific hardware in mind. The camera model was selected after inquiry within the faculty. The camera had been used successfully in a recent project, thus it had been demonstrated that it works well. The rest of the interface hardware (level converter and voltage converter) was selected after consulting with the person who had used the camera in a recent project.

5.2.3 Physical Implementation Costs

Table 5.1 provides an estimate of what it might cost to produce the device.

5.2.4 Camera Commands

The camera is controlled by sending command packets to the camera via a serial interface. The instructions are relayed with a simple command/acknowledge exchange between the camera and the controller [14]. A typical exchange is illustrated in figure 5.4.

Table 5.1: The estimated implementation costs.

Component	Cost
Camera	R500
Wavecom Q2686	R2000
GSM Triband Antenna	R600
Power Supply and Battery	R500
Enclosure and Lighting	R750
Assembly and Testing	R250
Total	R4600

5.2.5 Python Camera Driver

A driver for the camera was first written in Python to better understand the operation of the camera. It was also done to determine where potential problems might arise and to create proper unit tests for each camera command. Python code running on a computer is also much simpler to debug than code running on hardware.

There are three main camera commands Sync, Initialise and GetPicture.

- The Sync command is used to synchronise the camera with the controller. During the Sync command the camera detects the baud rate of the controller and once the rate has been detected the camera sends an acknowledge command.
- The Initialise command is used to define camera settings like the image size and resolution.
- The GetPicture command is used to tell the camera when to take a picture. The command also controls the transfer of the image from the camera to the controller.

Each command was tested by creating two virtual serial ports on a computer and then linking them. By linking the two ports, a connection between a device and a controller

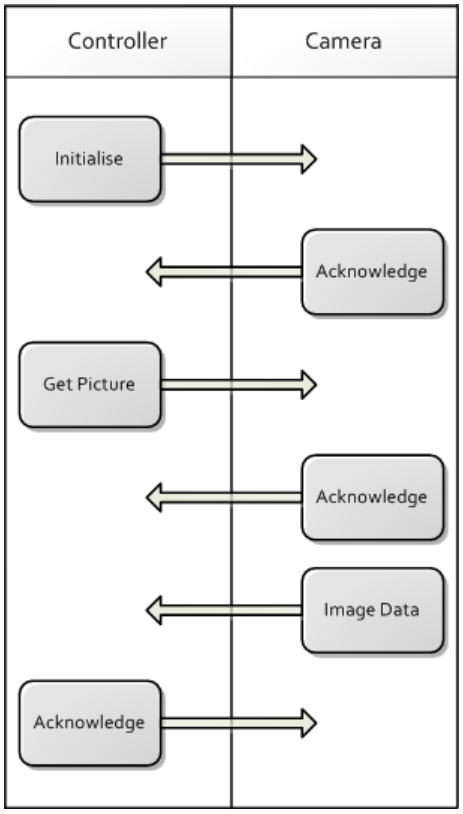


Figure 5.4: A typical command exchange between the controller and the camera.

is simulated. The driver code was then executed on the one port and the unit test code on the other. The final code was tested on the camera by connecting it to a computer's serial port and executing the driver software on the computer.

5.2.6 Wavecom Camera Driver

The operating system running on the Wavecom chip is event driven[15]. This means that one can subscribe to certain events and once the event happens, some specified code is executed. This complicates implementing driver code that requires waiting for specific responses from the device.

A Finite State Machine is a way of implementing a specific behaviour model. It consists of a finite number of states, with actions for moving between states specified[16].

To implement each commands, a Finite State Machine was used with different states representing each step where the driver has to wait for a response. The program code is supplied in Appendix A.

5.3 Histogram Modification

Histogram modification is performed in order to correct colour flaws in the meter images and to improve contrast. White areas may appear grey due to lighting or camera exposure. Colour correction changes these areas so that they are pure white.

OpenCV is used to perform the histogram modification with the 'Normalize' function. The function first calculates the histogram of the image. The range of this histogram is then determined and thus the location of the first bin and the last bin containing pixels. The image is then modified, so that the histogram range is stretched over all the histogram bins. This corrects the white colour flaws in the image and it improves the image contrast. Histograms of images before and after normalization are shown in figure 5.5 and figure 5.6.

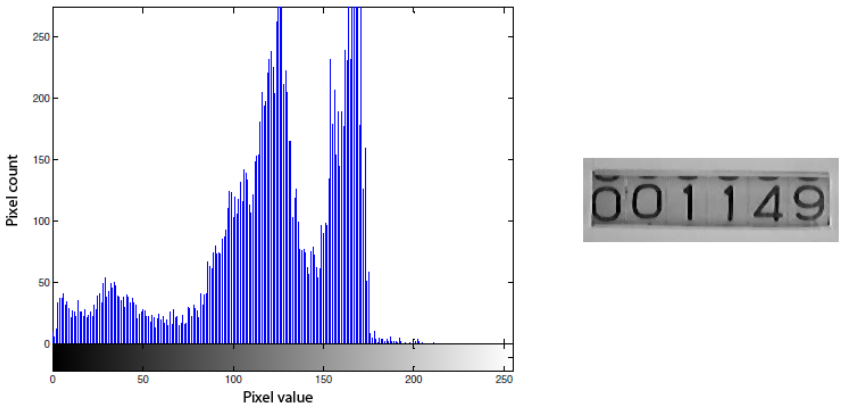


Figure 5.5: The image and image histogram before normalization.

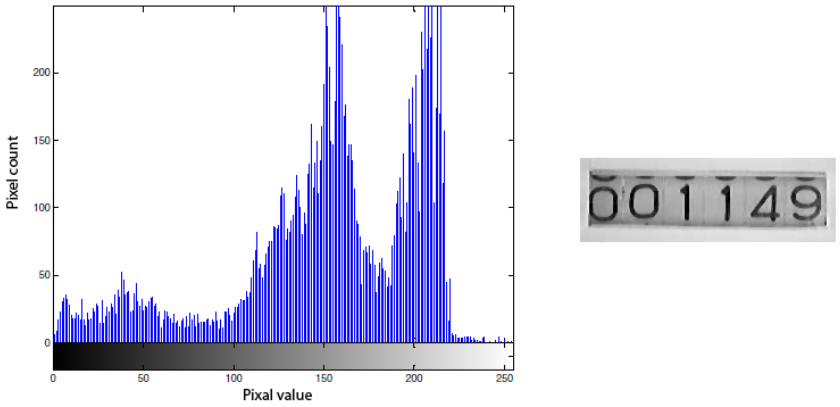


Figure 5.6: The image and image histogram after normalization.

5.4 Region Growing

In order to extract numbers from the images of bulk water meters, the coordinates of the numbers have to be known. Region growing is the process used to determine the coordinates of the rectangles bounding the numbers. Region growing is a seeded algorithm, which means that a start point has to be specified. Once a start pixel is specified, the coordinates of the region is determined through an iterative process. A region is defined as the area which contains all the dark pixels that are directly connected to the start point pixel.

5.4.1 Iterative Border Determination

In the region growing algorithm a process is repeated to determine the borders of the numbers. In the process the following values are specified:

- Centre pixel: This is a dark pixel that is already part of the region. The surrounding pixels are defined relative to the centre pixel.
- Initial pixel: This is a number between 1 and 8 and is used to specify at which neighbouring pixel the algorithm should start comparisons.

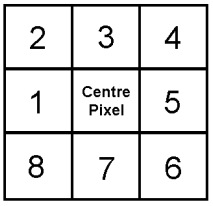


Figure 5.7: How pixels adjacent to the centre pixel are numbered.

The pixels adjacent to the centre pixel are numbered according to figure 5.7. When the process starts, the adjacent pixels are tested sequentially from the defined initial pixel to see whether the pixel is dark or not. Once a dark pixel is found, it is added to the region and the coordinates of the bounding rectangle are updated. For the next iteration, the dark pixel is used as the new centre pixel and the initial pixel value is recalculated. This process repeats itself until it reaches the original centre pixel.

5.4.2 Initial Pixel Specification

It is important to specify the pixel at which the process should start testing for dark pixels (initial pixel). If the process first inspects the centre pixel of the previous iteration, the process would move back to that pixel in the next iteration. Effectively the process would move back a step and the program could potentially become stuck in an endless loop. Incorrectly specifying the initial pixel could also result in an incorrect bounding rectangle being reported, if the whole number border isn't traversed.

Depending on where the start pixel is relative to the region, it is also important to specify the initial pixel. For example, if the pixel is located at the bottom of the region and the initial pixel is defined as the pixel directly above the centre pixel the process will move into the body of the region instead of around the border.

To determine the initial pixel of a new iteration, the position of the centre pixel of the previous iteration is determined relative to the centre pixel of the new iteration. The new

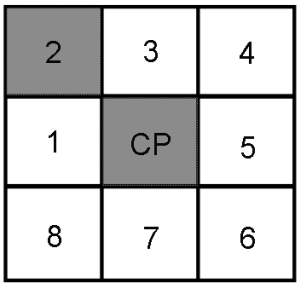


Figure 5.8: How the initial pixel is determined, the first step. CP indicated the centre pixel. The shaded blocks indicate where dark pixels have been found.

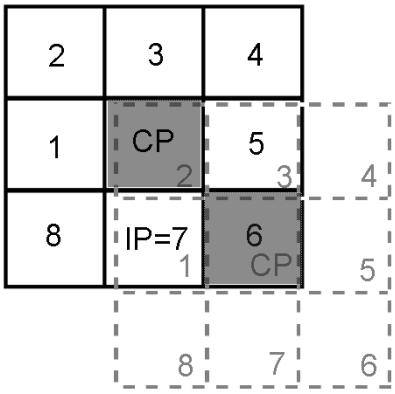


Figure 5.9: How the initial pixel is determined, the second step. The grey dashed grid indicates the previous step. IP indicates the new initial pixel.

initial pixel is specified as the pixel in the position sequentially following the centre pixel of the previous iteration. The two steps are illustrated in figure 5.8 and figure 5.9.

5.5 Centreline Approximation

In centreline approximation the coordinates of the vertical lines that pass through the centre points of the meter dials are calculated. The coordinates are used in the region

filtering process to determine the likelihood of a region corresponding to a number. To determine the lines, a list of unfiltered regions is used. The following steps are performed before the centrelines can be calculated:

- Firstly, the coordinates of the region centre points, the width of every region and the distance between regions centre points are calculated and stored in individual lists. See figure 5.10.
- From the list containing the distances between centre points, the median distance is calculated. This value will be used as the distance between centrelines.
- From the list containing the region widths, a median width is calculated. A region with the median width will be used as the starting point for calculating centreline coordinates.

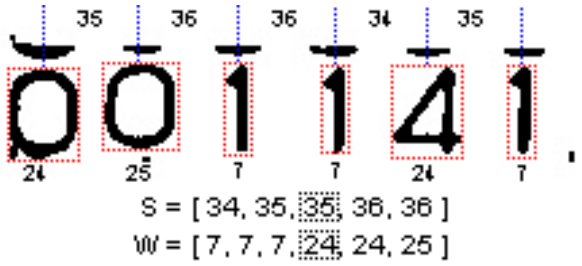


Figure 5.10: Image showing the spacing between regions (S) and the widths of regions (W). The lists of the values are sorted numerically with the median values indicated.

To calculate the coordinates of the centrelines, the starting point is used as the first coordinate. From there equally spaced coordinates are calculated to both sides of the starting point with the spacing equal to the median distance between centre points, as shown in figure 5.11. The coordinates are checked so that they do not exceed the borders of the image.

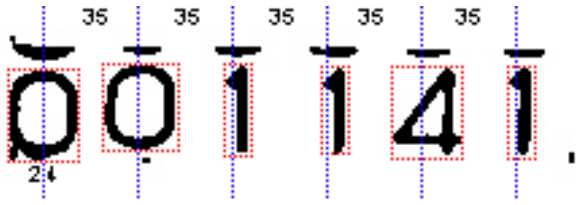


Figure 5.11: Centrelines are indicated on the image by the dashed vertical lines. The starting region width and the spacing between lines are displayed.

5.6 Region Filtering

Region growing is performed on two different horizontal lines on the image in order to ensure that no numbers or partial numbers are missed. Region filtering is the process of consolidating the two region lists and removing regions that correspond to noise. There are five main cases that occur in the filtering process that will be discussed next.

5.6.1 Case 1

In this case, one region growing line misses a number, because the number isn't centred in the dial window, as illustrated in figure 5.12. This can occur often, as the dials rotate gradually and do not change discretely.

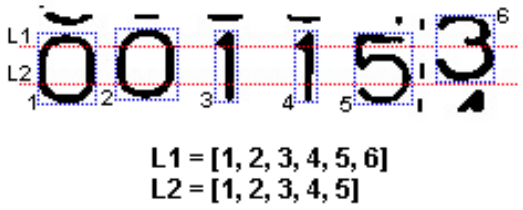


Figure 5.12: This image shows the meter dials after thresholding in case 1. The two region growing lines are illustrated as well as the regions found.

The solution is to compare the current entry of one list to the next entry of the other list. If the regions are the same, the region in the current entry of the second list is used as a starting point to search for other regions in the vertical plane via region growing. If another region is found, the starting region is modified to also contain the new region.

For clarity, this is explained with regards to figure 5.12. The lists in the figure 5.12 would have trailing zeros when implemented in software. When comparing entry 5 of both lists, they both contain region 5. In the next step, list 1 contains region 6, list 2 contains a zero. First, list 1's entry is compared to list 2's next entry which is zero. They do not match. Then list 2's entry is compared to list 1's next entry which is zero. They do match, so list 1's entry is used as a starting point for the region growing algorithm. The algorithm searches for other pieces in the vertical plane, and if it finds a piece it is added to the region in list 1. If the piece in list 1 is a whole number that is offset from centre (case 1), then no other regions are found, and the region is added to the final list as is. If the piece in list 1 is a half number (case 2), then the region growing algorithm finds the other piece, adds it to the existing region and the new region is added to the final list.

5.6.2 Case 2

In this case one region growing line misses one piece of a half-number, as illustrated in figure 5.13. This can happen when one region growing line passes through the space between one number and the next due to the way the dial is rotated.

The solution to this case is the same as for case 1.

5.6.3 Case 3

In this case, each region growing line finds a different region at the same approximate coordinates, as illustrated in figure 5.14. This happens when the one line finds one part of a half-number and the other line finds the other half.

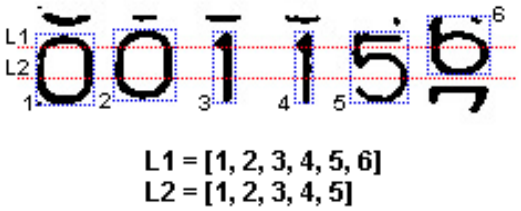


Figure 5.13: This image shows the meter dials after thresholding in case 2. The two region growing lines are illustrated as well as the regions found.

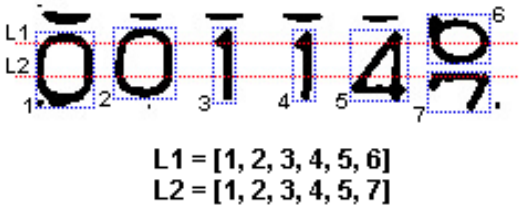


Figure 5.14: This image shows the meter dials after thresholding in case 3. The two region growing lines are illustrated as well as the regions found.

The solution is to merge the two regions. This is done by creating a new region that contains both regions.

5.6.4 Case 4

In this case the region growing lines find regions that are not numbers and can be considered as noise, as illustrated in figure 5.15. This can happen when the spaces between the dials are dark enough that they survive the thresholding step.

The solution is to calculate the area of a region. If the area is smaller than a specified value, the region is classified as noise and discarded.

Some regions may still have an area large enough to fail this test. For these cases all regions are compared against a list of coordinates that represent the centrelines of all the

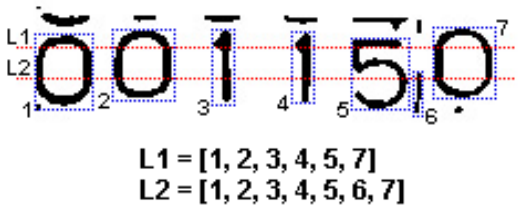


Figure 5.15: This image shows the meter dials after thresholding in case 4. The two region growing lines are illustrated as well as the regions found.

meter dials. If a centreline does not pass through the region the region is discarded.

5.6.5 Case 5

In this case a shadow is included in the region thereby distorting the number, as illustrated in figure 5.16. This can happen when shadows on the meter face extend over a number. During thresholding the number and the shadow is joined in one dark region.

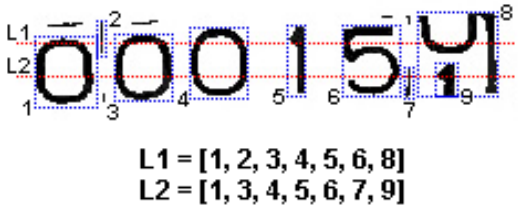


Figure 5.16: This image shows the meter dials after thresholding in case 5. The two region growing lines are illustrated as well as the regions found.

The solution is to compare regions against a list of coordinates that represent the centrelines of all the meter dials. If a region is wider than half a dial width from a centreline, the region coordinates are adjusted to fall within the distance.

5.7 Conclusion

In this chapter the main components of the program were explained in detail. In the next chapter the problems encountered during the implementation will be discussed.

Chapter 6

Implementation Problems

6.1 Introduction

This chapter will discuss the steps involved in reaching the final implementation of program components and methods. The objective is to show which methods were implemented and evaluated before reaching the final implementation. It will also explain why certain methods were discarded. The relevant program components will be listed and all methods implemented will be discussed.

6.2 Rectangle Detection

6.2.1 Dial Window Detection

Initially an OpenCV rectangle detection program was used to try and find the border of the dials window. This proved unsuccessful as the window rectangle did not stand out sufficiently from the rest of the meter face.

6.2.2 Black Border Detection

In order to make the dials window more visible, a black border was placed around it. This improved the results greatly, but there were still cases where the border did not stand out enough for the program to find the black rectangle border.

6.2.3 High Contrast Border Detection

The border was changed from a black rectangle to a white rectangle with a black outline. This provided a much higher contrast border to the dials window. Rectangle detection with the high contrast border has a nearly 100% success rate.

6.3 Image Enhancement

6.3.1 Smoothing

During implementation it had to be determined where in the program it would be most effective to apply the smoothing algorithm. Through visual inspection and experimentation, it was decided that the smoothing algorithm should be applied before the threshold step, as it results in the best number integrity.

6.3.2 Threshold and Histogram Modification

6.3.2.1 Visual Inspection I

Initially images were used that were not taken in a controlled constant, environment. This caused difficulty in finding a threshold value that would work well for all images. In order to find an ideal threshold value for each image, histogram peak inspection was evaluated.

6.3.2.2 Histogram Peak Inspection

The images of meter dials consist of mainly very light and very dark pixels. This would cause the image histograms to have two peaks; one for the dark pixels and one for the lighter pixels. In order to find an ideal threshold value, the image histogram was inspected to find the lowest point between the two peak values, which was then used for the threshold value. This method has two issues; the bin size selection is important and the image contrast should be high. If the bin sizes are too small local minimum values are found instead of the minimum between the two peaks. If the bin sizes are too large an inaccurate value is determined for the threshold value. Images with a low contrast do not always have two well defined peaks in the histogram. If the images are too dark with low contrast, there is only one peak in the histogram, in which case the method fails.

6.3.2.3 Histogram Modification

In order to increase image contrast, histogram modification was performed. The 'Normalize' function in OpenCV was used to modify the histogram. The function determines the positions of the first and last non-zero histogram bins. It then scales the histogram so that the bins are in the first and last positions in the histogram, thus there are no initial and trailing zero bins. This method worked very well in increasing image contrast. Histograms of images before and after normalization are shown in figure 5.5 and figure 5.6.

6.3.2.4 Visual Inspection II

In order to improve the evaluation of all test procedures, images were taken in a fixed setup with constant lighting as can be seen in section 7.3. This ensured that images taken are very consistent in term of contrast and light exposure. This eliminated the need to find individual threshold values for each image, as a universal value could now easily be determined for all images.

6.4 Character Extraction

6.4.1 Extraction

6.4.1.1 Area Division

The first method used to determine the coordinates of the six dial numbers was area division. The position of the first and the last pixel in the image was determined and the distance between them divided by six. This provided the width of the dials from which the dial border coordinates were calculated. This method was not very accurate and varying number widths significantly affected the results. In order to determine the number coordinates with greater accurately, the Radon Transform was investigated.

6.4.1.2 Radon Transform

The radon transform was suggested in discussions with an expert in the signal processing field. After consulting a source[17], it seemed to be a valid method to persue.

The radon transform works by determining the sum of all vertical pixel values for each horizontal pixel position[17]. In the results, continuous blocks of high values correspond to the regions on the image that contain the numbers. Thus coordinates can be determined from the high value regions which can be used to extract numbers from the image. This only worked for some numbers. It did not fare well with numbers that do not have many pixels over the whole area of the number. A '0' for instance has a low pixel count over the centre of the number and 0's were frequently cut in half by the process like in figure 6.1.

6.4.1.3 Region Growing

Region growing consists of specifying an initial pixel that is part of a region and then inspecting neighbouring pixels to determine whether to add them to the region or not.

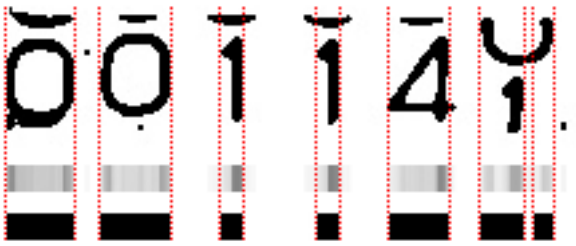


Figure 6.1: Example of number extraction with the radon transform. The grey bar is the result of the radon transform. The darker areas represent higher pixel counts. The black bar represents the regions extracted with the radon transform. The dashed lines indicate where the image will be cut to extract the regions.

For this application, the program starts at the far left of the image at a certain height and inspects each pixel until the far right is reached. When a dark pixel is found, it is defined as the start pixel to the region growing algorithm in order to determine a region. Once the far right of the image has been reached, the program returns a list of regions. Since the program steps through pixels at a specific height, it is possible to miss numbers if the line passes above or below a number or between two numbers.

6.4.1.4 Double Region Growing

In order to prevent some numbers being missed due to the height on which the pixels are inspected, the algorithm is run on two different heights. This ensures that all the numbers are found between the two instances of the algorithm. The different regions from the two instances are then merged in the 'Filtering' step.

6.4.2 Region Growing Algorithm

6.4.2.1 Constant Initial Pixel

In the first version of the algorithm, the initial pixel value was always the same, in other words, the pixel inspection would always start from the same relative pixel. This caused the problems that were explained in section 5.4.2.

6.4.2.2 Variable Internal Initial Pixel

To prevent the algorithm from moving into the body of the number, the algorithm was modified so that the initial pixel value is recalculated in every cycle of the algorithm. As such it would be taken into account from which direction the algorithm came in order to not put the program into a loop scenario.

6.4.2.3 Variable First Initial Pixel

At this stage when the algorithm was first executed, the first initial pixel value was fixed. Since the start point for the algorithm can be on any side of the number it is also important to be able to specify the first initial pixel value. The algorithm was modified to include this functionality.

6.4.3 Filtering

The problems encountered during the implementation of the filtering process have already been discussed in the Detailed Work chapter.

6.5 Conclusion

In this chapter the methods implemented before reaching a final implementation for all relevant program components were discussed.

In the next chapter the effectiveness of the program will be evaluated through a series of tests.

Chapter 7

Experimental Investigation

7.1 Introduction

The objective of this chapter is to determine how accurate the program can extract dial readings from meter images and how effective parts of the program are. This will be done by capturing images of the meter dials in various positions, running them through the complete program and evaluating the results.

7.2 Motivation

In order to determine the effectiveness of the program it is necessary to perform several tests. Tests will be performed for situations for which the character recognition software has been trained and for situations for which it has not been trained. With these results the program can be evaluated and it can be determined whether the program is suitable for the intended application. Tests will also be performed on individual program components in order to determine how effective they are.

7.3 Setup

The experiments are performed with the camera mounted a fixed distance away from the meter face. Two lights are used to evenly illuminate the dials. Light diffusing material is placed between the lights and the meter in order to reduce glare on the meter face from the lights. The camera is connected to a computer via a serial interface and the computer is used to issue camera commands and retrieve images. The camera is powered by the Wavecom development board. Figure 7.1 shows the device setup. The meter used in these experiments is the same meter that was used for capturing images for the training data.

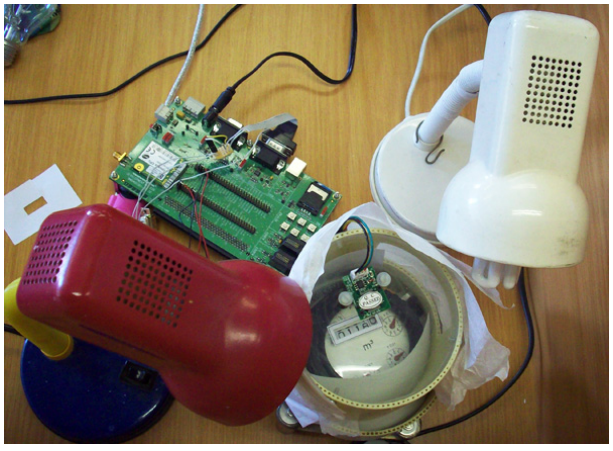


Figure 7.1: The device setup for capturing images.

In practice permanent sources of illumination, such as LED's, should be implemented. The experiments are performed on the same meter that was used for training Tesseract. These types of bulk water meters have a protective cover to shield the glass from scratches, thus the glass from another meter should not provide optical interference. For the testing, new images were captured, thus no training images were reused. The final product will also conform to a standardized setup.

7.4 Test 1: Median Filter

7.4.1 Execution

A set of 20 images were used for the test. First the images were run through the whole program, then the median filter component was removed and the program run again. The amount of images successfully processed were examined as well as the quality of the images returned in each case.

7.4.2 Results

The results of the two tests were examined visually. The images that were not subjected to the median filter had significantly more noise present in the images. The difference can be better observed after the images have been thresholded. Figure 7.2 shows a thresholded image where a median filter was applied in the previous step, figure 7.3 shows a thresholded image where the median filter was not applied in the previous step. In the thresholded images it can also be seen that the numbers are less degraded when the median filter was applied before thresholding. Due to the increase in noise, the region growing algorithm had trouble isolating the individual numbers. Out of the 20 images, number could be extracted successfully from 20 images in the case where the median filter was present, but numbers could only be extracted successfully in 5 images in the case where the median filter was not present.

Figure 7.2: A thresholded image where the median filter was applied in the previous step.

Figure 7.3: A thresholded image where the median filter was not applied in the previous step.

7.4.3 Conclusion

From these results it can be deduced that the median filter plays a vital role in the program in order to successfully extract readings from the meter images.

7.5 Test 2: Histogram Normalisation

7.5.1 Execution

A set of 20 images were used for the test. First the images were run through the whole program, then the histogram normalisation component was removed and the program run again. The amount of images successfully processed were examined as well as the quality of the images returned in each case.

7.5.2 Results

With the histogram normalisation component included, 20 out of 20 images were processed successfully, without the component 19 out of 20 images were successfully processed.

7.5.3 Conclusion

The histogram normalisation component does not appear to be critical to the functioning of the program.

7.6 Test 3: Tesseract Accuracy I

7.6.1 Execution

Images will be captured with the dials at various stages of rotation. Test cases for the following situations will be performed and evaluated:

- Whole numbers; only single numbers visible on the dials.
- Half-numbers; some dials contain two numbers where 50% of each number is visible.
- Half-numbers; some dials contain two numbers where 75% of one number is visible and 25% of the other.

For every case, one image will be captured of each possible number. It was felt that one image would be sufficient to show whether a number can be recognised or not. This is because each dial is identical, and with a constant setup, extracted digits of the same number in the same position will always look almost identical. The motivation for this stemmed from observing the results of the test cases. When observing the other dials in the image (not the furthest right dial), it was observed that the same digit is repeated in many images with no recognition problems occurring over multiple cases.

The images will be subjected to the enhancement techniques discussed in the Implementation Outline chapter and character recognition will be performed on the resultant images. The character recognition results will be verified for accuracy and the overall accuracy will be determined.

7.6.2 Results

7.6.2.1 Case 1

Case 1 tested only whole numbers, thus where 100% of each number was visible. Ten samples were used in the testing process. Tesseract was trained for whole numbers.

In table 7.1 the test image samples from case 1 are displayed along with the Tesseract output and correct results. Ten out of 10 samples were correctly identified.

7.6.2.2 Case 2

Case 2 tested only half-numbers where 50% of each number was visible. Ten samples were used in the testing process. Tesseract was trained for these types of half-numbers.

In table 7.2 the test image samples from case 2 are displayed along with the Tesseract output and correct results. Ten out of 10 samples were correctly identified.

7.6.2.3 Case 3

Case 3 tested only half-numbers where 75% of the bottom number was visible and 25% of the top number. Ten samples were used in the testing process. Tesseract was not trained for these types of half-numbers.

In table 7.3 the test image samples from case 3 are displayed along with the Tesseract output and correct results. Three out of 10 samples were correctly identified.

7.6.2.4 Case 4

Case 4 tested only half-numbers where 75% of top number was visible and 25% of the bottom number. Ten samples were used in the testing process. Tesseract was not trained for these types of half-numbers.

In table 7.4 the test image samples from case 4 are displayed along with the Tesseract output and correct results. Seven out of 10 samples were correctly identified.

7.6.3 Interpretation

In all cases the image enhancement performed well resulting in clear numbers to pass to Tesseract for character recognition. In case 1 and case 2 where Tesseract had been trained

Table 7.1: Results from case 1. Dial images and character recognition outputs are displayed.

Original Dials	Reconstructed Image	Tesseract Output	Correct Answer
	0 0 1 1 4 1	001141	001141
	0 0 1 1 4 2	001142	001142
	0 0 1 1 4 3	001143	001143
	0 0 1 1 4 4	001144	001144
	0 0 1 1 4 5	001145	001145
	0 0 1 1 4 6	001146	001146
	0 0 1 1 4 7	001147	001147
	0 0 1 1 4 8	001148	001148
	0 0 1 1 4 9	001149	001149
	0 0 1 1 5 0	001150	001150

Table 7.2: Results from case 2. Dial images and character recognition outputs are displayed.

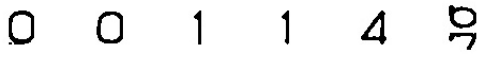
Original Dials	Reconstructed Image	Tesseract Output	Correct Answer
		00114a	00114a
		00116b	00116b
		00114c	00114c
		00114d	00114d
		00114e	00114e
		00114f	00114f
		00114g	00114g
		00114h	00114h
		00116i	00116i
		0011gj	0011gj

Table 7.3: Results from case 3. Dial images and character recognition outputs are displayed.

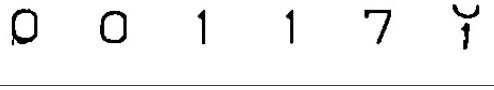
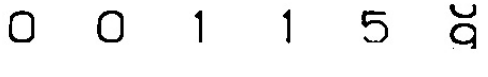
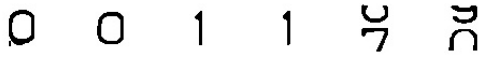
Original Dials	Reconstructed Image	Tesseract Output	Correct Answer
		00117a	00117a
		00116h	00116b
		00116j	00116c
		00115i	00115d
		00115j	00115e
		00115j	00115f
		00116i	00116g
		00116ic	00116h
		00115i	00115i
		0011gj	0011gj

Table 7.4: Results from case 4. Dial images and character recognition outputs are displayed.

Original Dials	Reconstructed Image	Tesseract Output	Correct Answer
		00116a	00116a
		00116b	00116b
		00115c	00115c
		00116d	00116d
		00116e	00116e
		001162	00116f
		00115i	00115g
		00116h	00116h
		00116e	00116i
		0011gj	0011gj

for the type of numbers observed, the results were very good with 100% recognition for the provided examples. In case 3 and case 4 where Tesseract had not been trained for the sample numbers, the overall recognition success was 50%. Thus Tesseract performed well where the digits are similar to the training data but it started to fail rapidly when the digits were significantly different.

In the application it would not be possible to capture images that only contain dials in positions that Tesseract has been trained for. Considering these results it might be advantageous to train Tesseract for a wider variety of possible dial positions in order to get better overall character recognition results.

7.7 Test 4: Tesseract Accuracy II

7.7.1 Execution

In this test the feasibility of training Tesseract for cases where 75% of one number and 25% of another number are visible is investigated.

Initially images were captured of the dials in the 75% / 25% positions in order to provide training data for Tesseract. It proved difficult to capture images with the dials in consistent posistions due to large changes in visibility from slight changes in dial positions. In order to overcome this sensitivity problem, it was decided to modify the program behaviour when numbers that are approximately in the 75% / 25% position, are encountered. The program was modified to discard the smaller part and to only keep the larger part.

The same images used for case 3 and case 4 of the previous test will be used in this test. Images will be subjected to the modified program and character recognition performed on the resultant images. The character recognition results will be verified for accuracy to determine the overall accuracy.

7.7.2 Results

7.7.2.1 Case 3 version 2

Case 3 tested only half-numbers where 75% of the bottom number was visible and 25% of the top number. Ten samples were used in the testing process.

In table 7.5 the test image samples from case 3 version 2 are displayed along with the Tesseract output and correct results. Seven out of 10 samples were correctly identified.

7.7.2.2 Case 4 version 2

Case 4 tested only half-numbers where 75% of top number was visible and 25% of the bottom number. Ten samples were used in the testing process.

In table 7.6 the test image samples from case 4 version 2 are displayed along with the Tesseract output and correct results. Seven out of 10 samples were correctly identified.

7.7.3 Interpretation

During the test it was found that training Tesseract for 75% / 25% numbers proved difficult. The modified program produced an increase in recognition accuracy from 50% to 70%, for 75% / 25% numbers. This increases the overall recognition accuracy to 85%. The modification to the filtering process involved discarding the smaller piece when a 75%/25% number is encountered. The recognition accuracy increased because the 75% part resembles a whole number closer that a 75%/25% number resembles a half number.

Considering these results, it might be better to focus on analysing the character recognition outputs than to try to increase Tesseract recognition accuracy.

Table 7.5: Results from case 3 version 2. Dial images and character recognition outputs are displayed.

Original Dials	Reconstructed Image	Tesseract Output	Correct Answer
	0 0 1 1 7 1	001171	001171
	0 0 1 1 6 2	001169	001162
	0 0 1 1 6 3	001163	001163
	0 0 1 1 5 4	001154	001154
	0 0 1 1 5 5	001155	001155
	0 0 1 1 5 6	00115i	001156
	0 0 1 1 6 7	001167	001167
	0 0 1 1 6 8	001168	001168
	0 0 1 1 5 9	001150	001159
	0 0 1 1 7 g	0011gj	0011gj

Table 7.6: Results from case 4 version 2. Dial images and character recognition outputs are displayed.

Original Dials	Reconstructed Image	Tesseract Output	Correct Answer
	0 0 1 1 6 U	001160	001160
	0 0 1 1 6 1	001161	001161
	0 0 1 1 5 2	00115i	001152
	0 0 1 1 6 3	001163	001163
	0 0 1 1 6 4	001164	001164
	0 0 1 1 6 5	001160	001165
	0 0 1 1 5 6	001155	001156
	0 0 1 1 6 h	00116h	00116h
	0 0 1 1 6 8	001168	001168
	0 0 1 1 g j	0011gj	0011gj

7.8 Conclusion

In this chapter it was shown that characters for which Tesseract has been trained for, are accurately recognized. Recognition accuracy is low for other characters resulting from other dial positions. Training Tesseract for other dials positions proved difficult, but good results were achieved by modifying the program. In the next chapter the document will be summarised and the content critically discussed.

Chapter 8

Conclusion

8.1 Conclusion

In this thesis a prototype device was designed and built that can capture images and transmit them to a server via the GSM network. The device was mounted on a bulk water meter, in order to capture images of the meter dials.

A program was developed that enhances the captured images and extracts the numbers on the meter dials. The program then constructs a new image from the extracted numbers in order to produce an image suitable for character recognition.

Character recognition software, Tesseract, was found that could be trained for a new character set. Tesseract was trained with images of whole numbers and half-numbers. It was found that Tesseract has high recognition accuracy for images that contain characters similar to the training data, but accuracy drops significantly in other cases.

Through investigation it was found that in cases where Tesseract has not been trained, better recognition results were achieved by modifying the program as detailed in the Experimental Investigation chapter. The software was evaluated and it was shown that an overall recognition accuracy of 85% can be achieved with the program modifications as set out. In the case of whole- and half-numbers, accuracies of 100% were achieved in

both instances. In other cases accuracies of 70% were achieved.

8.2 Overview of the Project and Operational Contributions

The following is an overview of the project as well as the main operational contributions of this thesis:

- A cost effective image capturing and transmission telemetry system was developed, retrofittable to a standard bulk water meter.
- A CMOS based digital camera was integrated into the telemetry system and the necessary driver software developed.
- A survey was done to determine which open-source software would be suitable and convenient for the character recognition requirements of the project and an initial evaluation process was carried out on the selected software.
- Image enhancement algorithms were subsequently developed to improve the performance of the character recognition software.
- A custom region growing algorithm was developed specifically suited to the application.
- Training of the character recognition tool was carried out in parallel to the algorithm development.
- The effectiveness of the enhancements and the training process was continuously evaluated until a point was reached where it was felt that further improvements in accuracy would be better achieved by other non character recognition based methods.

- An overall recognition accuracy of 85% was achieved.
- The feasibility of the approach was demonstrated beyond a doubt and indicated a way forward to actual application.

8.3 Future Work

The work conducted in this thesis could form an integral part of a viable operational setup. The following future work is recommended:

- Although a feasible prototype was constructed, industrialisation is required. A field mechanical design should be carried out, taking into account lighting and battery requirements.
- Currently the device software only captures images on request. The current device software can be expanded to include a schedule for capturing images. A minimum capturing schedule should be determined in order to utilize the inherent heuristic information from the meter dials. Images acquired can then be analysed with the methods developed in this project and the readings extracted. The readings can then be statistically analysed by utilising the historical information inherent in the sequential images in order to extract the correct readings and to eliminate the problem of partially visible dial numbers.
- A suitable database structure and interface should be developed in order to store the images and relevant meter information. A GUI should also be developed in order to relay the information to the users.

Appendices

Appendix A

Program Code (CD)

List of References

[1] Endress & Hauser: Proline Promag 10H Technical Information[Online]. Available at: http://www.za.endress.com/, [2010, November], 2010.

[2] Cross Instrumentation: Meinecke OPTO Pulser Data Sheet[Online]. Available at: http://www.meineckemeters.com/forms/ds_pulser.pdf, [2010, November], 2009.

[3] I. T. L. Education Solutions Limited: *Introduction to Computer Science*. Dorling Kindersley, 2004.

[4] Schulenburg, J.: GOCR[Online]. Available at: http://jocr.sourceforge.net/, [2010, November], 2010.

[5] Unknown: Ocrad[Online]. Available at: http://www.gnu.org/software/ocrad/, [2010, November], 2009.

[6] Smith, R.: Tesseract[Online]. Available at: http://code.google.com/p/tesseract-ocr/, [2010, November], 2010.

[7] Smith, R.: An Overview of the Tesseract OCR Engine. In: *Proc 9th Int. Conf. on Document Analysis and Recognition*, pp. 629–633. 2007.

[8] Rice, S., Jenkins, F. and Nartker, T.: The Fourth Annual Test of OCR Accuracy. Tech. Rep., University of Nevada, Las Vegas, July 1995.

[9] Smith, R.: Tesseract[Online]. Available at: `http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract2`, [2010, November], 2010.

[10] ABBYY: ABBYY FineReader Product Page[Online]. Available at: `http://finereader.abbyy.com`, [2010, November], 2010.

[11] Congitive Technologies: News - OCR CuneiForm[Online]. Available at: `http://www.cognitive.ru/news/30/`, [2010, November], 2007.

[12] Mather, P.M.: *Computer processing of Remotely Sensed Images: An Introduction*. John Wiley and Sons, 2004.

[13] Burger, W. and Burge, M.: *Digital Image Processing: An Algorithmic Introduction Using Java*. Springer, 2008.

[14] COMedia: COMedia C328 UART Camera Module[Online]. Available at: `http://www.seeedstudio.com/depot/uart-camera-module-with-jpeg-compression-c328-p-209.html`, [2010, November], 2010.

[15] Sierra-Wireless: Sierra Wireless Airprime Product Information[Online]. Available at: `http://www.sierrawireless.com/`, [2010, November], 2010.

[16] Robinson, J.A.: *Software Design for Engineers and Scientists*. Newnes, 2004.

[17] Astola, J.: *Advances In Signal Transforms: Theory and Applications*. Hindawi Publishing Corporation, 2007.